IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON MOBILE EDGE COMPUTING AND MOBILE CLOUD COMPUTING: ADDRESSING HETEROGENEITY AND ENERGY ISSUES OF COMPUTE AND NETWORK RESOURCES

Received April 10, 2019, accepted April 29, 2019, date of publication May 2, 2019, date of current version May 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2914450

Fog-Assisted Aggregated Synchronization Scheme for Mobile Cloud Storage Applications

GIWON LEE[1], HANEUL KO[2], SANGHEON PACK[3], VALENTINO PACIFICI[4], AND GYÖRGY DÁN[4]

[1]Samsung Electronics, Suwon 16677, South Korea
[2]Department of Computer Convergence Software, Korea University, Sejong 30019, South Korea
[3]School of Electrical Engineering, Korea University, Seoul 02841, South Korea
[4]School of Electrical Engineering, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden

Corresponding author: Sangheon Pack (shpack@korea.ac.kr)

This work was supported in part by the National Research Foundation of Korea through the Korean Government under Grant NRF-2017R1E1A1A01073742, and in part by the Swedish Foundation for Strategic Research (SSF) through the Modane Project under Grant NRF-2014R1A2A1A12066986.

ABSTRACT Cloud storage applications, such as Dropbox and Google Drive, have recently become very popular among mobile users. In these applications, a cloud server is responsible for synchronizing updates to files among mobile users, and thus if files are shared by many mobile users and are frequently updated then the resulting synchronization traffic can be significant. In order to reduce the synchronization traffic with providing acceptable access latency, we propose a fog-assisted aggregated synchronization (FAS) scheme in which the fog computing server and the cloud server conduct localized and aggregated synchronizations, respectively. We develop an analytical model of the FAS scheme based on renewal-reward theory and use it for model-based adjustment of the timer that controls the trade-off between access latency and synchronization traffic. We use analytical and simulation results to give insight into the effects of the timer, the update-to-access ratio, the number of mobile users, and the sensitivity to the arrival process. The analytical and simulation results demonstrate that the FAS scheme can reduce the synchronization traffic significantly with acceptable access latency compared to conventional schemes.

INDEX TERMS Mobile cloud storage, synchronization traffic, fog-assisted aggregated synchronization, renewal-reward theory.

I. INTRODUCTION

Cloud storage applications have become very popular among mobile users in recent years [2]–[6]. As an example, Dropbox, one of the most popular cloud storage applications, has more than 100 million users worldwide, and allows its users to access and share their files from mobile devices in real-time. As another example, Microsoft announced that they will soon enable editing documents from Android tablets, so that mobile users can access Office files and tools while on the go [7], [8].

Most cloud storage applications require or encourage their users to install a designated client program and to assign a designated local folder. A mobile user can add or modify a file in the local folder and the update is then automatically synchronized with a cloud server (CS). When the file is shared with other mobile users, the CS is responsible for synchronizing the shared file with all sharing users. Such a centralized architecture may be easy to manage, but if the file is shared by many mobile users and is frequently updated then the synchronization traffic can be significant [9], [10]. For instance, Li *et al.* [11] analyze a recent large-scale Dropbox trace collected at the ISP level [12] and reveal that the synchronization traffic contributes to more than 90% of the traffic in Dropbox. Even with geo-distributed systems (e.g., Microsoft OneDrive), Wu *et al.* [13] show that complex synchronization protocols lead to non-negligible performance overhead. Since excessive synchronization traffic in mobile networks is critical owing to limited radio resources and battery power [14], reducing the synchronization traffic is one of the most important issues in mobile cloud storage applications.

The existing synchronization schemes for cloud storage applications fall into on of two categories [15]: update-triggered and timer-triggered. In the update-triggered synchronization scheme, once an update occurs, the full content

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Shuja.

of the file is updated to the CS and is synchronized with mobile users. Update triggered synchronization can be easily implemented and provides strong consistency, i.e., the file used by mobile users is always up-to-date, but it leads to significant synchronization traffic since the file is synchronized upon each update. In the timer-triggered synchronization scheme there is a pre-defined timer, and once the timer expires, the updated file is delivered to the CS and is synchronized to mobile users. Subsequently, a new timer is set for the next synchronization. Compared with the update-triggered synchronization, the timer-triggered synchronization scheme can reduce the synchronization traffic but it does not ensure strong consistency until the timer expires.

These two schemes correspond to extremes in terms of the trade-off between synchronization traffic and consistency. Recent work has aimed at addressing this trade-off through applying one or the other scheme to a file depending on the importance of ensuring consistency [21], thus allowing to manage the trade-off for a set of files. The problem of managing the trade-off between synchronization traffic and consistency for each individual file is however so far unsolved. The problem of managing the trade-off is even more intriguing when file access patterns are heterogeneous in space, as the best scheme to use depend on the frequency of file accesses and may hence be location specific.

To address this problem, in this paper we propose a fog-assisted aggregated synchronization (FAS) scheme to reduce the synchronization traffic for files that are frequently updated and are shared by many mobile users. In the proposed FAS scheme, mobile users are grouped into domains depending on their geographical locations and each domain is managed by one fog server (FS). Since the FS can be co-located with an adjacent access point (AP)/base station (BS), the FS can synchronize mobile users within its domain with little traffic.[1] In the FAS scheme, when an update occurs in the domain of a FS, the update is first sent to the FS. The FS synchronizes the update with mobile users in its domain and sends the update to the CS, which informs the other mobile users about the existence of an update. The CS accumulates updates during a timer T. When the timer expires, the CS delivers aggregated updates to the other FSs (except to the originating FS) in order to synchronize mobile users in the domains of the other FSs. Note that the CS sends aggregated updates to the FSs instead of to the individual mobile users, and the FS can multicast aggregated updates to mobile users in its domain.

Owing to localized and aggregated synchronization, the FAS scheme has the potential to reduce the synchronization traffic significantly. Since the CS informs mobile users in the other domains about the existence of updates to particular files, mobile users can request aggregated updates from the CS before the timer expires if needed, which allows to satisfy strong consistency [18]. Due to requesting aggregated updates, the FAS scheme may introduce additional latency when accessing a file, thus the timer T has to be chosen to guarantee an acceptable access latency, while reducing the synchronization traffic. To find such a timer value, we design a lightweight timer selection algorithm based on an analytical model of the FAS scheme. Extensive simulation results are given to demonstrate the effects of the timer, the update-to-access ratio, the number of mobile users, and the sensitivity to the arrival process. The analytical and simulation results show that the FAS scheme can reduce the synchronization traffic with acceptable access latency compared to conventional schemes.

The main contributions of this paper are twofold. First, the proposed FAS scheme is a novel solution to a fundamental problem in mobile cloud storage synchronization management, i.e., the reduction of the synchronization traffic by using a fog architecture, and as such it can be widely used in mobile cloud storage applications. Second, we develop an analytical model of the synchronization traffic and of the access latency of the proposed FAS scheme, and we validate the model by extensive simulations. The presented analytical and simulation results provide valuable insights for further performance improvements of not only the FAS scheme, but for the management of cloud storage synchronization solutions in general.

The rest of this paper is organized as follows. Related works are summarized in Section II. The system model and the UDS and TDS schemes are introduced in Section III, and the fog-assisted aggregated synchronization scheme is described in Section IV. The analytical model is developed in Section V. Simulation results are presented in Section VI, followed by concluding remarks in Section VII.

II. RELATED WORK

Many mobile cloud storage applications for data sharing have been widely deployed and gained much popularity in recent years. As a result, it is expected that cloud storage applications will generate a large amount of mobile Internet traffic [12]. Specifically, Li *et al.* [9] reveal that frequent and short updates to files in Dropbox generate an excessive amount of maintenance traffic, despite a number of techniques already in use for traffic reduction.

A common technique used by cloud storage applications for reducing network traffic is delta encoding. Delta encoding is an effective way for reducing the synchronization traffic. Drago *et al.* [10] compare five popular cloud storage applications and demonstrate that Dropbox is able to reduce the synchronization traffic significantly by using delta encoding.

Data compression is another way to reduce the synchronization traffic. Li *et al.* [11] reveal that 52% of files in cloud storage applications can be effectively compressed and data compression is able to reduce 24% of the total synchronization traffic based on a real-world trace.

[1] This architecture is analogous to the cloudlet architecture [17] where the cloudlet is located between mobile users and the CS, and takes on several tasks from mobile users to reduce the end-to-end response time and the load of the CS. Unlike the existing cloudlet architecture, the proposed architecture focuses on the synchronization among mobile users instead of task offloading.

Server data deduplication eliminates replicas on the storage server [19], [20]. Similarly, when the same content is already present in the CS, replicas in the users' local folder can be identified to save synchronization traffic. Drago *et al.* [10] show that Dropbox does not upload the data if the data are readily available at the CS. Moreover, Dropbox can identify multiple copies of users' files even after they are deleted and later restored.

Nonetheless, the synchronization traffic of cloud storage applications is significant even when the above techniques are employed and thus a number of research efforts have tried to further reduce the synchronization traffic. Li *et al.* [9] develop an update-batched delayed synchronization scheme that acts as a middleware between a Dropbox folder and an actual folder on the file system. By means of batching updates from the actual folder and applying them to the Dropbox folder at once, the synchronization traffic can be significantly reduced. In another work [15], they propose a timer-based synchronization mechanism. Every time the timer expires, batched updates are synchronized and the timer is recalculated to consider the latest inter-update time. Ramos *et al.* [21] propose a scalable cloud file sharing system, which considers users' consistency requirements on the shared file while avoiding unnecessary synchronization traffic. In their system, updates requiring strong consistency are rapidly synchronized to users whereas other updates are delayed and batched to reduce the synchronization traffic. E *et al.* [22] designed an efficient cross-cloud file collaboration system called CoCloud. Specifically, they developed an inter-cloud transfer protocol including adaptively chunked deduplication, adjusted compression, and multi-level bundling. Cui *et al.* [23] proposed QuickSync to reduce the synchronization traffic. In their system, a network-aware chunker to adaptively select the propose chucking strategy based on network conditions, a redundancy eliminator to correctly perform delta encoding, and a batched syncer to appropriately transmit chunks in a batch manner and reduce the number of reconnections are introduced. In our previous work [24], we propose an efficient delta synchronization algorithm that aggregates the update to reduce the synchronization traffic and synchronizes aggregated updates periodically to satisfy strong consistency. To identify the optimal policy for the aggregation and the periodical synchronization, an optimization problem is formulated as a Markov decision process (MDP).

These previous works do not consider a fog architecture for mobile cloud storage applications. The fog architecture is compatible with the emerging mobile edge cloud (MEC) architecture [25], [26], in which cloud computing and storage resources located close to the network edge can be used to provide application specific services and have access to radio network information. Our proposed FAS scheme leverages the MEC architecture to reduce the synchronization traffic through a combination of periodic and on-demand synchronization. Since cloud-based collaboration through mobile cloud storage applications is typically used in geographically confined areas, such as a company office or a university laboratory [27], the proposed FAS architecture, which leverages geographical proximity, is a promising solution to reduce the synchronization traffic.

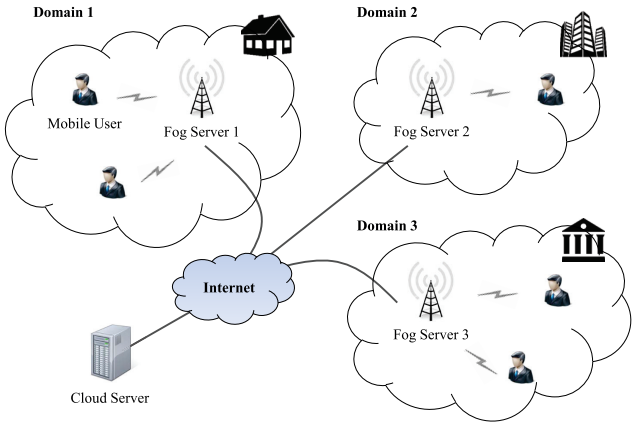

FIGURE 1. System model illustrating FS domains ($D = 3$).

III. SYSTEM MODEL AND BACKGROUND

In this section, we describe the system model and give an outlines of the UDS and the TDS schemes.

A. SYSTEM MODEL

Figure 1 shows the system model of the FAS scheme. We consider a mobile network, in which there is one FS per domain co-located with an AP/BS. FS domain can be a company office or a university laboratory where many mobile users use cloud storage applications. We assume that mobile users are connected to one FS through a wireless link, whereas FSs are connected to the CS through a mobile backhaul. We consider a single file F shared by mobile users and synchronized by the CS. By using localized and aggregated synchronizations, FS is responsible for synchronizing mobile users within its domain. We denote by D the number of FS domains and by M_d the number of mobile users in the FS domain d ($1 \leq d \leq D$). In Figure 1, we assume that there are 3 FS domains.

To avoid simultaneous updates to file F, only one user is granted the right to update the file F at a time [18]. We refer to the domain in which this user is located by D_1, and call it the originating domain. This is a reasonable simplifying assumption for systems where individual files are modified by a single user over an extended period of time, and it is also reasonable if files are segmented into chunks and access control is managed at the chunk level (in this case our notion of a file would correspond to a chunk). The delegation of the right to update can be solved using existing protocols via the CS. We consider that delta encoding is used for efficiency, and we denote by u_j the jth update of file F. We consider multicast capable mobile networks, and since the FS maintains the list of mobile users in its domain, it can multicast updates to mobile users in its domain. To describe the operation of the FAS scheme, we define four messages.

- **Update**(F, u_j): This message is sent to the FS and to the CS to inform them about the jth update of F.
- **Sync**(F, u_j, u_{j+1}, ...): The FS and the CS send this message to synchronize updates (i.e., u_j, u_{j+1}, ...) with mobile users. Note that the CS can aggregate multiple updates to minimize the synchronization traffic.
- **Notify**(F): This message is sent by the CS to notify mobile users in FS domains about an available update of file F (except in the originating domain). By receiving the message, mobile users can know that there is an update to F and thus they should contact the CS upon access to F. To avoid duplicating this message, the CS sends this message only when the first update to F occurs after the last synchronization of mobile users in the other domains.
- **Access**(F): The mobile user sends this message to access the shared file F after receiving a **Notify**(F) message.

B. BACKGROUND

Before describing the proposed FAS scheme, we provide an overview of the UDS and the TDS schemes. In the UDS scheme, whenever an update occurs, the CS synchronizes the update to all mobile users. The message flow of the UDS scheme is illustrated in Figure 2. In the figure U_k stands for mobile user k, and U_1 is a mobile user that makes updates to F. When U_1 first updates the file at time τ_1, an **Update**(F, u_1) message is delivered to the CS. Then, the update u_1 is delivered to all users by sending **Sync**(F, u_1) messages immediately to all users (U_2, U_3, and U_4 in the figure). The same procedure is followed for u_2 and for u_4 at times τ_2 and τ_4, respectively. As a result, mobile users can access the up-to-date file F immediately when there is an access event (e.g., U_4 at time τ_3).

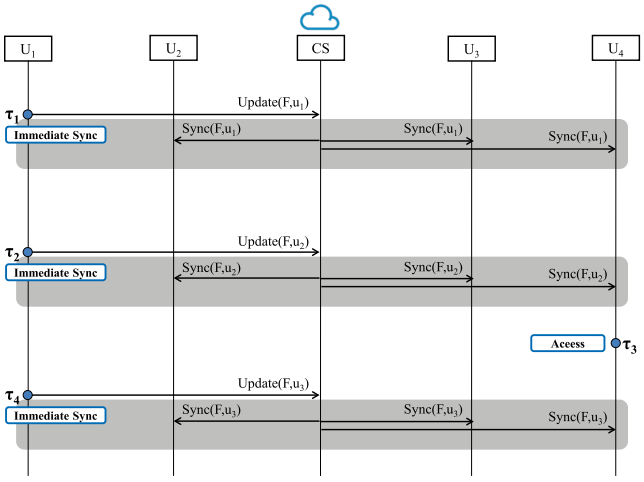

FIGURE 2. Operation of update-triggered delta synchronization (UDS) scheme.

In the case of the TDS scheme, the CS aggregates updates during a pre-defined timer. The message flow of the TDS scheme is illustrated in Figure 3. When U_1 first updates the file at time τ_1, an **Update**(F, u_1) message is sent to the CS,

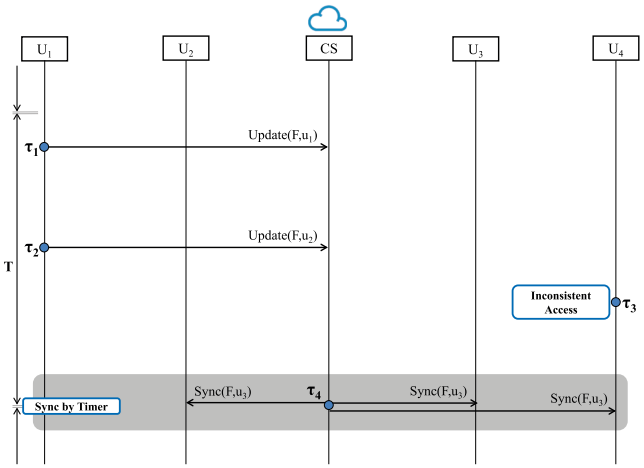

FIGURE 3. Operation of timer-triggered delta synchronization (TDS) scheme.

which then aggregates u_1. Similarly, the next update u_2 at time τ_2 is also aggregated in the CS. U_3 can not access the up-to-date file F at time τ_3 because the timer has not yet expired. When the timer expires at time τ_4, aggregated updates including u_1 and u_2 are synchronized by means of **Sync**(F, u_1, u_2) messages.

If frequent updates occur and the number of mobile users sharing the file is large, the synchronization traffic of the UDS and the TDS schemes can be significant. Moreover, the TDS scheme does not provide consistency after an update until the timer expires.

IV. FOG-ASSISTED AGGREGATED SYNCHRONIZATION (FAS) SCHEME

In this section, we describe the operation of the FAS scheme, whose main objective is to reduce the synchronization traffic in mobile cloud storage applications. To achieve this goal, the FS is used for localized synchronization while the CS performs aggregated synchronization, as follows. If an update occurs, the update is synchronized to other mobile users within the originating domain D_1 by the FS, after which the FS sends the update to the CS. To reduce the synchronization traffic, the update is not synchronized immediately to mobile users in the other domains. Instead, the CS sends a **Notify**(F) message to the FSs, except to the originating FS, and the FSs deliver the **Notify**(F) message to mobile users in their own domains. As a result, all mobile users in the other domains will know that there is an update available for F, even though they have not received the update itself.

The CS aggregates updates during the timer T and sends to the FSs aggregated updates using **Sync**(F, u_j, ...) messages when the timer expires. Each FS can then multicast aggregated updates to mobile users in its domain. If a mobile user wants to access F after receiving the **Notify**(F) message but before the timer expires, the mobile user contacts the CS by sending an **Access**(F) message, which will cause the CS to send **Sync**(F, u_j, ...) messages. On the other hand, if a mobile user has not received any **Notify**(F) message,

it can use the local copy of F, as it was synchronized by the most recent **Sync** message.

Intuitively, the FAS scheme may lead to additional latency to access F compared with the UDS scheme. Therefore, an appropriate timer should be chosen to guarantee an acceptable expected access latency as well as reduced synchronization traffic. We will address this problem in Section V-C. In what follows, we illustrate the operation of the FAS scheme through two examples.

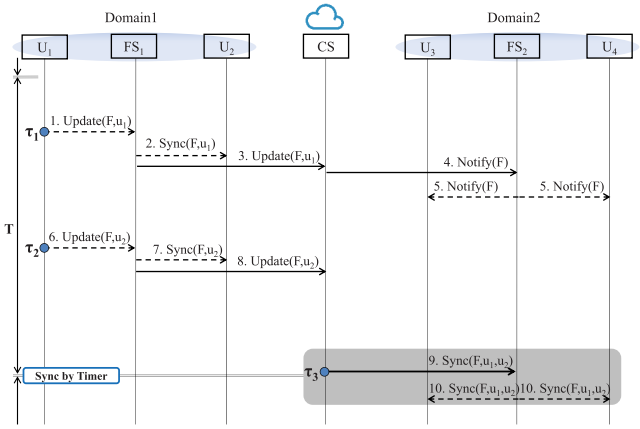

FIGURE 4. Example: Synchronization due to timer expiration.

A. SYNCHRONIZATION DUE TO TIMER EXPIRATION

Figure 4 shows the operation of the FAS scheme when the synchronization occurs due to the expiration of the timer. U_1 sends an **Update**(F, u_1) message to FS_1 at time τ_1 (Step 1).[2] If there are mobile users sharing F in the domain of FS_1 (e.g., U_2), FS_1 multicasts a **Sync**(F, u_1) message to them (Step 2). After that, FS_1 sends an **Update**(F, u_1) message to the CS (Step 3). Note that the CS does not send any **Sync**(F, u_1) message to the other FSs immediately to keep the synchronization traffic low, but it sends a **Notify**(F) message to mobile users in the other domains to inform them about the existence of the update to F (Steps 4-5). The same happens upon the update u_2 at time τ_2, which is synchronized to mobile users only within the originating domain by FS_1 but it is not sent to mobile users in the other domains because the update of F has been already notified owing to the update u_1 (Steps 6-8). The CS aggregates two updates, and upon the timer expiration at time τ_3, it sends aggregated updates by **Sync**(F, u_1, u_2) messages to all FSs except to the originating FS_1 (Steps 9-10). Note that the CS sends aggregated updates to the FSs instead of to the individual mobile users because each FS can multicast aggregated updates to mobile users, e.g., FS_2 can multicast the **Sync**(F, u_1, u_2) message to U_3 and U_4. Through multicasting, the gain of the FAS scheme increases with the number of mobile users per domain sharing the same file.

[2]Since only one user is granted the right to update file F at a time, the CS receives Update message from one domain.

B. SYNCHRONIZATION DUE TO ACCESS OF MOBILE USER

Figure 5 shows the operation of the FAS scheme when the synchronization is performed due to the access request of a mobile user in the other domains. There is an update u_1 of F at time τ_1 (Step 1). Then, FS_1 multicasts a **Sync**(F, u_1) message to mobile users in its domain (e.g., U_2), and sends an **Update**(F, u_1) message to the CS (Steps 2-3). The CS aggregates the update and sends **Notify** messages to the other FSs, which the other FSs deliver to mobile users in their domains, since this is the first update after the last synchronization (Steps 4-5). U_2 can access F immediately at time τ_2 without contacting the CS because U_2 is in the originating domain. Nonetheless, when U_4 in another domain wants to access F at time τ_3, it sends an **Access**(F) message to the CS via FS_1 because it received the **Notify**(F) triggered by u_1 at time of τ_1 (Steps 6-7). In response, the CS synchronizes the file by sending the **Sync**(F, u_1) message to all FSs, and it restarts the timer (Steps 8-9). Note that U_4 would be able to access file F without contacting the CS if no **Notify**(F) message had been received after the last synchronization.

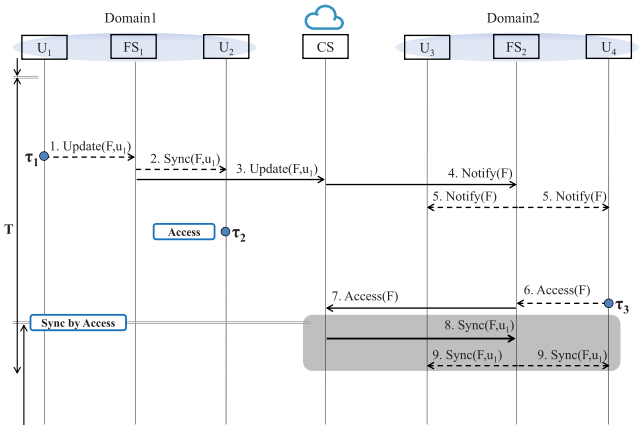

FIGURE 5. Example: Synchronization due to access(F) message.

TABLE 1. Frequently used notation.

T	Value of the timer
M_d	Number of mobile users in domain d
D	Number of FS domains
S_s	Size of **Sync** message without any update
S_u	Size of a single update
S_n	Size of **Notify** message
S_a	Size of **Access** message
X_n	Length of renewal period n

V. ANALYTICAL MODEL

In this section, we derive an analytical model of the synchronization traffic and of the access latency of the FAS scheme. Table 1 contains the most frequent notation used in the model.

To facilitate the analysis, we consider that the access events follow a Poisson process with intensity λ_a [28], [29], update events follow a Poisson process with intensity λ_u [30], and the two processes are independent. We denote the arrival intensity of access events outside of domain D_1 by $\lambda_{a'} = \left(1 - \frac{M_1 - 1}{M - 1}\right)\lambda_a$, where M_1 denotes the number of mobile

users in the originating domain and M represents the total number of users. We use the notation $N_A(t)$, $N_{A'}(t)$, and $N_U(t)$ for the number of access events, access events outside of domain D_1, and update events during a period of length t, respectively. These are Poisson distributed random variables, and we define the events $A(t)$, $A'(t)$, and $U(t)$ that there are access event, access event outside of domain D_1, and update event at time t. Finally, we define the update-to-access ratio $\rho = \lambda_u/\lambda_a$ for file F, which shows the relative frequency of the two kinds of events.

We denote by S_s the size of a **Sync** message without any update, and by S_u the size of a single update. For simplicity we assume that each update has size S_u, but we note that in principle S_u could be interpreted as the average size of an update, assuming that the size of an update is independent of the inter-update time. Thus, if n_u updates are aggregated and synchronized with a **Sync** message, the size of the **Sync** message will be $S_s + n_u S_u$. Finally, we denote the size of the **Notify** message and the **Access** message by S_n and S_a, respectively.

To compute the average synchronization traffic, we model the system as a renewal-reward process. The renewal epochs are the **Sync** messages sent by the CS to the FSs, and consequently to all mobile users. The renewal epoch could be triggered by a timer expiration or by an **Access** message sent by a mobile user in a domain different from D_1, if there was an update since the last **Sync** message. We denote the number of renewal events up to time t by $N_R(t)$, and we let $R(t)$ be the event that there is a renewal event at time t. We denote by t_n the beginning of renewal period n, and by $X_n = t_{n+1} - t_n$ its length. For a particular renewal period, we denote the time of the kth access event by t_k^a, the time of the kth access event outside of the originating domain by $t_k^{a'}$, and the time of the kth update event by t_k^u.

A. SYNCHRONIZATION TRAFFIC

The reward C_n during renewal period n is the synchronization traffic, i.e., the traffic generated by **Update**, **Notify**, and **Sync** messages during the interval $(t_n, t_{n+1}]$, and the average synchronization traffic is

$$C = \lim_{t\to\infty} \frac{1}{t} \sum_{n=1}^{N_R(t)} C_n. \tag{1}$$

Let us consider the synchronization traffic C_n consists of the synchronization traffic C_n^1 incurred in the originating domain where updates occur and the synchronization traffic C_n^o incurred in the other domains, i.e.,

$$C_n = C_n^1 + C_n^o. \tag{2}$$

In the originating domain, synchronization is performed whenever an update occurs, and since updates are not aggregated, the traffic due to a single synchronization is given by $S_s + S_u$. The expected synchronization traffic is thus $E[C_n^1] = \lambda_u(S_s + S_u)E[X_n]$.

To derive the synchronization traffic C_n^o in the other domains, we have to consider the synchronization traffic due to the **Sync** message, due to the **Notify** message sent in response to the first update after a **Sync** message, and due to the **Access** message.

Observe that the renewal periods can end by one of two reasons: due to a timer expiration or due to an access event in the other domains. A renewal period can end due to a timer expiration, in which case $t_{n+1} - t_n = T$. This is the case if no update event happens on $(t_n, t_n + T]$, if there is no access event outside of domain D_1 on $(t_n, t_n + T]$, or if there are one or more access events outside of domain D_1 on $(t_n, t_n+T]$ but all updates happen after the last access event of the period.

We can use the law of total probability to compute the probability that a renewal period ends due to a timer expiration. By conditioning on the number of access and update events during a period of length T, we obtain

$$P(X_n = T) = \sum_{n_{a'}=0}^{\infty} \sum_{n_u=0}^{\infty} P(t_{n_{a'}}^{a'} < t_1^u | n_{a'}, n_u) \times P(N_{A'}(T) = n_{a'}, N_U(T) = n_u). \tag{3}$$

To compute the probability $P(t_{n_{a'}}^{a'} < t_1^u | n_{a'}, n_u)$ that all access events happen before the first update event, observe that due to the Poisson arrival assumption, the distribution of the arrival epochs conditional on the number $n_{a'}$ of access events during a period of length T is uniform on $[0, T]^{n_{a'}}$ [31]. Given that there are $n_{a'}$ access events and n_u update events during a period of length T, we can thus compute the probability that the first update event happens after the last access event using combinatorial arguments as

$$P(t_{n_{a'}}^{a'} < t_1^u | n_{a'}, n_u) = \binom{n_{a'} + n_u}{n_{a'}}^{-1}. \tag{4}$$

We can also express the expected number of update events during a renewal period that ends due to a timer expiration as

$$E[N_u(X_n) | X_n = T] = \sum_{n_{a'}=0}^{\infty} \sum_{n_u=0}^{\infty} P(t_{n_{a'}}^{a'} < t_1^u | n_{a'}, n_u) \, n_u P(N_{A'}(T) = n_{a'}, N_U(T) = n_u), \tag{5}$$

which together with the size of the **Sync** messages allows the computation of the expected synchronization traffic during a renewal period that ends due to a timer expiration

$$E[C_n^o | X_n = T] = (S_n + S_s)P(N_u(X_n) > 0 | X_n = T) + S_s P(N_u(X_n) = 0 | X_n = T) + S_u E[N_u(X_n) | X_n = T]. \tag{6}$$

Alternatively, a renewal period can end due to an access event in the other domains, in which case $X_n = t_{n+1} - t_n < T$. This is the case if there is at least one update since the last **Sync** message before an access event happens outside of domain D_1. Clearly, $P(X_n < T) = 1 - P(X_n = T)$, but to be able to compute the expected length of a renewal period, we need to compute the density function $f_{X_n}(t)$, $t \in (0 < t < T)$ of the renewal period length. For this,

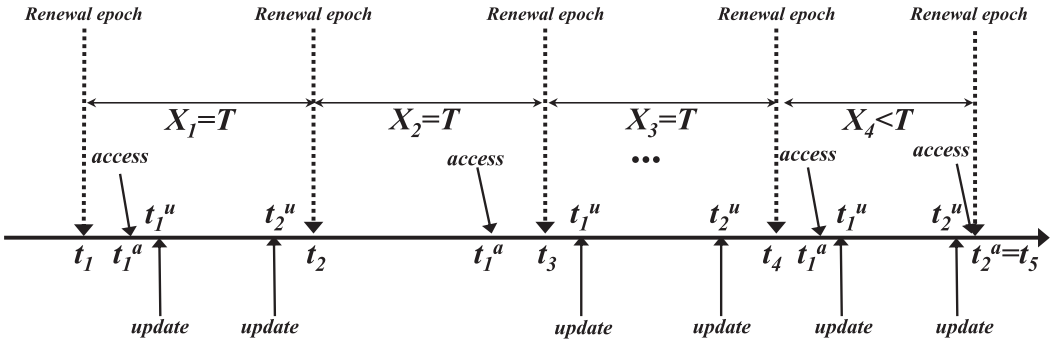

FIGURE 6. Illustration of 4 renewal periods, periods 1-3 end due to timer expiration, period 4 ends due to an access event.

we again use the law of total probability, conditioning on the number of access events up to and including t and on the number of update events up to t. Observe that given that there are $n_{a'}$ access events and n_u update events, the renewal period ends after time t if the $n_{a'}$th access event outside of D_1 happens at time t and the first $n_{a'}-1$ access events happened before the n_u update events, as otherwise the renewal period would have ended earlier. The probability that the $n_{a'}$th access event happens at time t is given by the Erlang($n_{a'}$) distribution with density function $f_E(t, n_{a'}, \lambda_{a'})$ while the probability that the first $n_{a'}-1$ access events happened before the first of the n_u update events for $n_{a'}>0$ and $n_u>0$ is

$$P(t^{a'}_{n_{a'}-1} < t^u_1 | n_{a'}, n_u, t_{n_{a'}} = t) = \binom{n_{a'}-1+n_u}{n_{a'}-1}^{-1}, \quad (7)$$

and thus we have

$$f_{X_n}(t) = \sum_{n_{a'}=1}^{\infty} f_E(t, n_{a'}, \lambda_{a'}) \sum_{n_u=1}^{\infty} P(N_U(t)=n_u) \binom{n_{a'}-1+n_u}{n_{a'}-1}^{-1}. \quad (8)$$

Based on $f_{X_n}(t)$ we can compute the expected renewal period length due to an access event as

$$E[X_n | X_n < T] = \int_0^T t f_{X_n}(t) dt. \quad (9)$$

To compute the traffic, observe that there is always one **Notify** and one **Access** message sent out during a renewal period that ends due to an access event. The expected number of updates during the renewal period can be computed by taking the expectation of (8) with respect to n_u,

$$E[N_u(X_n) | X_n < T] = \int_0^T \sum_{n_{a'}=1}^{\infty} f_E(t, n_{a'}, \lambda_{a'}) \sum_{n_u=1}^{\infty} n_u P(N_U(t)=n_u) \binom{n_{a'}-1+n_u}{n_{a'}-1}^{-1} dt, \quad (10)$$

which together with the size of the **Sync** messages allows the computation of the expected synchronization traffic in the other domains during a renewal period that ends due to an access event as

$$E[C^o_n | X_n < T] = (S_n + S_s + S_a) + S_u E[N_u(X_n) | X_n < T]. \quad (11)$$

Finally, combining (3) with (6) and with (11) provides us the total synchronization traffic.

Based on the above, we can calculate the average traffic over time as

$$C = \lim_{t\to\infty} \frac{1}{t} \sum_{n=1}^{N_R(t)} C_n = \frac{E[C_n]}{E[X_n]}. \quad (12)$$

where the equality holds with probability one according to the renewal reward theorem [31].

The computation of the average traffic C using (12) requires the numerical evaluation of (3) and (8), which contain infinite sums. Nonetheless, as the terms in (3) and (8) decrease exponentially in $n_{a'}$ and n_u, using a partial sum for the evaluation introduces an error that can be bounded.

B. ACCESS LATENCY

We can use the above model to express the average latency L of accessing file F. Since all updates are synchronized by the FS in the originating domain, the access latency in the originating domain is $L^1 = 0$.

Unlike users in the originating domain, mobile users in the other domains may need to send an **Access** message to access file F. This is the case if there are aggregated updates in the CS upon an access event. Thus, to compute the average access latency, we need to compute the probability that an access event would lead to the generation of an **Access** message, and we need to compute the time L^o it takes until the corresponding **Sync** message is received.

We compute the probability that an arbitrary access event triggers a **Sync** message by using the elementary renewal theorem to compute the expected rate of renewal epochs

$$\lim_{t\to\infty} \frac{1}{t} N_R(t) = \frac{1}{E[X_n]}, \quad (13)$$

and we observe that only a fraction $P(X_n < T)$ of the renewal epochs is triggered by an access event. Thus, the probability

that an access event outside of FS domain D_1 triggers a renewal epoch can be expressed as

$$P(R(t)|A(t)) = P(X_n < T)\frac{1}{\lambda_{a'}E[X_n]}. \quad (14)$$

To be able to compute the expected time it takes to receive the **Sync** message, we need to compute the expected size of the **Sync** message, which is a function of the number of updates $N_u(X_n)$ since the start of the renewal period. The average number of updates since the beginning of the renewal period until an access event that triggers a renewal epoch is $E[N_u(X_n)|X_n < T]$, which can be computed based on (10) and (3), and provides the expected size of the **Sync** message. The average access latency can then be computed as $L = P(R(t)|A(t))L^o$.

C. MODEL-BASED TIMER SELECTION UNDER LATENCY CONSTRAINT

The synchronization traffic can be significantly reduced by increasing T, as a high value of T allows more updates to be aggregated at the CS. Nonetheless, a larger value of T implies increased access latency. In what follows, we propose a lightweight timer selection algorithm which ensures that the average access latency does not exceed a target threshold L_B. The algorithm is based on the following observation.

Proposition 1: The average access latency L is a monotonically increasing function of the timer T.

Proof: Let us first observe that the expected number of updates in a renewal period that ends due to an access event (cf. (10)) is monotonically increasing with the timer T, hence so is the size of a **Sync** message. Let us now consider the probability $P(R(t)|A(t))$ expressed in (14). By using the quotient rule for computing the derivative of (14) with respect to the timer T, it is easy to see that $P(R(t)|A(t))$ is also monotonically increasing with the timer T. To provide an intuition for the result, recall that it is only access event that happen after the first update event of the renewal epoch that trigger a renewal epoch. Furthermore, if we consider the remaining time $T - t_1^u$ until the next timer expiration following the first update event, then it is easy to see that for $T' > T$, the random variable $T' - t_1^u$ first-order stochastically dominates $T - t_1^u$. Since the remaining time increases with the timer T, there is a higher probability that an access event happens before the timer expires (and thus triggers a renewal), and thus the fraction of access events that trigger a renewal increases. ■

Proposition 1 allows us to formulate a simple algorithm for choosing a timer that minimizes the synchronization traffic subject to an average access latency constraint for a measured arrival intensity, based on the model. Let us denote by τ the largest timer value that satisfies $L \leq L_B$ (Step 1), i.e.,

$$\tau = \max\{T : L \leq L_B\}. \quad (15)$$

Intuitively, τ is the timer with lowest overhead. Nonetheless, observe that for $T < 1/\lambda_u$ the synchronization due to timer expiration would occur more frequently than update events and thus the synchronization traffic would be unnecessarily high. Therefore, the FAS scheme should only be used if $\tau \geq 1/\lambda_u$, in which case the optimal timer value is $T = \tau$. Otherwise, the CS should not aggregate updates and synchronization should be done immediately upon updates (i.e., just like UDS); we use $T = 0$ to denote this case. Algorithm 1 shows the pseudo-code of the timer selection algorithm.

Algorithm 1 Pseudo-Code of the Timer Selection Algorithm

Model-based timer selection algorithm

```
Let τ = max{T : L ≤ L_B}
if τ ≥ 1/λ_u then
    Let T = τ
else
    Let T = 0 /* timer disabled */
end if
```

VI. PERFORMANCE EVALUATION

In what follows, we present numerical results based on the analytical model and simulation results using an event-driven simulator written in C++. The presented results serve for three purposes: 1) to validate the analytical model; 2) to compare the performance of the FAS scheme with that of the UDS and the TDS schemes; and 3) to show that the timer selection algorithm is fairly insensitive to the arrival process. In the simulation, the events of **Update**(F, u_j), **Sync**$(F, u_j, u_{j+1}, \dots)$, **Notify**(F), and **Access**(F) are generated and the corresponding synchronoization cost and latency are computed. To obtain a realistic update-to-access ratio ρ, we use Dropbox data sets collected during 42 consecutive days reported in [12]. Based on the data sets, for a value of $\lambda_u = 1$ we use a value of $\lambda_a = 0.416$, i.e., $\rho = 2.4$. We consider that mobile users are uniformly spread among FSs (i.e., $M_1 = M_2 = \cdots = M_D$). In simulations, the default value of the acceptable access latency L_B is set to 30 *msec*.

We simulated domains that consist of a single-hop wireless link with bandwidth W and are connected to the CS by H-hop wireline backhaul with bandwidth B, and we compute the access latency based on the transmission time $L^o = (H/B + 1/W)(S_s + n_u S_u)$.[3] We consider that the cost of traffic over the wireless access link may be different from that over the backhaul [32], and thus we allow to weight the traffic over the wireless access link by a factor ω. Thus, if a **Sync** message with a single update is sent from the FS to mobile users over the wireless access link, the traffic cost is $\omega\,(S_s + S_u)$. While this network model may be rudimentary, it is simple and it allows us to focus on the fundamental characteristics of the FAS scheme.

The timer of the TDS scheme is set to that of the FAS scheme in simulations to allow for a fair comparison. The default parameter values for simulations are derived from [11], [33] and are summarized in Table 2. The presented

[3]The value of dividing the file size by the data rate cannot represent accurately the transmission time due to the randomness of the wireless channel, but we believe that it can be an approximate and simple model of transmission time for wireless links.

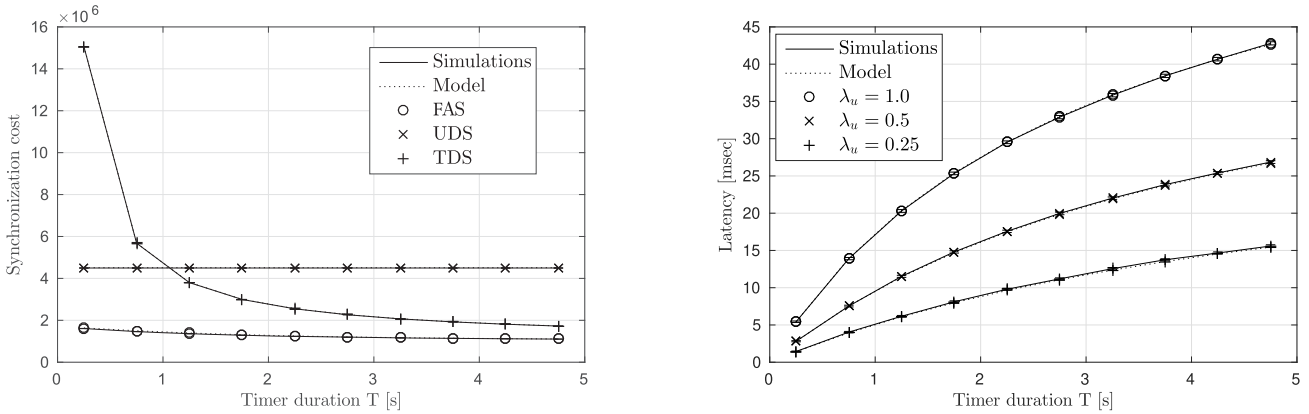

FIGURE 7. Synchronization cost and access latency vs. timer T ($\lambda_a = 0.416$, $M_d = 3$). (a) Synchronization cost. (b) Access latency.

TABLE 2. Default parameter values for simulations.

Parameter	S_s	S_u	S_n	S_a	ω
Value	40 $Kbytes$	10 $Kbytes$	60 $bytes$	60 $bytes$	10
Parameter	B	W	H	D	M_d
Value	100 $Mbps$	54 $Mbps$	10	3	3

simulation results are the averages of 100 simulation runs, and the figures show the 95% confidence intervals.

A. EFFECT OF TIMER (T)

Figure 7(a) shows the synchronization cost as a function of the timer T. Naturally, the synchronization cost of the UDS scheme is constant regardless of T because the UDS scheme does not employ any timer for the synchronization. On the other hand, the synchronization costs of the FAS and the TDS schemes decrease as T increases. The difference is significant for small values of the timer T, which shows the efficiency of the proposed FAS scheme in decreasing the synchronization traffic. It is also worthwhile to note that unlike the TDS scheme, the proposed FAS scheme always leads to less overhead than the UDS scheme.

Figure 7(b) shows the access latency of the FAS scheme as a function of the timer T. Note that the access latency of the UDS scheme is 0, and so is that of the TDS scheme, but the TDS scheme does not provide consistency. The figure shows that the access latency of the FAS scheme is a concave increasing function of the timer in accordance with Proposition 1. Comparing the results for different update intensities, we can also observe that the access latency decreases slower than linear in the update intensity.

Based on Figure 7, we can conclude that the analytical and simulation results for FAS overlap, which validates the model. Thus in the following, we show analytical results only, unless otherwise noted.

Figure 8 shows the trade-off between the access latency and the synchronization cost for five combinations of access intensity and update intensity, obtained by varying the timer T. The figure shows that the trade-off is fairly

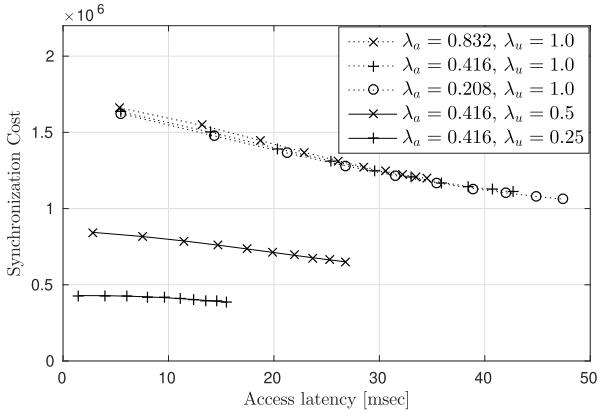

FIGURE 8. Synchronization cost vs. access latency trade-off ($M_d = 3$).

insensitive to the access intensity, but the update intensity has a significant impact. The significant impact of the update intensity on the trade-off curve is apparent from Figure 8. A closer inspection of the curves for different access intensity values shows that while the access latency and synchronization cost both change depending on the access intensity (a lower access intensity results in lower synchronization cost but higher access latency for a fixed value of T), the tradeoff curves are fairly similar. Thus, similar access latency-synchronization cost combinations can be achieved by adjusting the timer length T.

B. EFFECT OF THE UPDATE-TO-ACCESS RATIO (ρ)

Figure 9 shows the synchronization cost as a function of the update-to-access ratio ρ for timer $T = 2$ and constant access intensity $\lambda_a = 0.416$. While the synchronization cost of all three schemes increases linearly with ρ, it is clear from the figure that the proposed FAS scheme outperforms both the UDS and the TDS schemes, and the performance difference increases with ρ, i.e., as the arrival intensity of updates increases, in accordance with the corresponding curves in Figure 8.

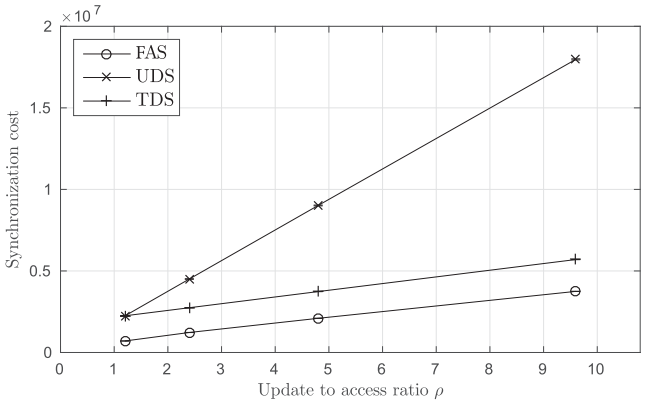

FIGURE 9. Synchronization cost vs. update-to-access ratio ρ ($T = 2$, $M_d = 3$).

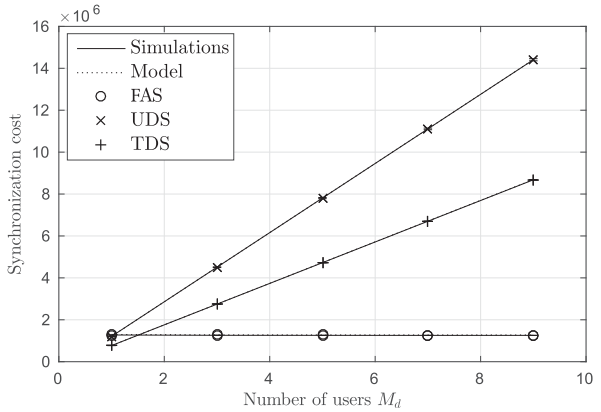

FIGURE 10. Synchronization cost vs. number of users M_d ($\lambda_a = 0.416$, $\lambda_u = 1$, $T = 2$).

C. EFFECT OF USER POPULATION (M_d)

Figure 10 shows the synchronization cost as a function of the number of mobile users per domain, M_d, for $T = 2$. The synchronization cost of the FAS scheme is constant regardless of M_d, because the FS multicasts updates within its domain without additional overhead. Consequently, the gain of the FAS scheme compared to the UDS and the TDS schemes increases as M_d increases. Although it may seem unfair to compare schemes employing unicast and multicast transmission, it is important to note that it is the inclusion of a fog server in the FAS system architecture that enables the use of multicasting in each domain, hence the comparison.

D. SENSITIVITY TO THE ARRIVAL PROCESS

Finally, we evaluate the sensitivity of the results to the arrival process using simulations. As alternatives to the Poisson process (i.e., exponential inter-arrival time), we consider that the inter-arrival times of update and access events follow one of two Phase-type distributions: Erlang-k and hyper-exponential. To keep the mean inter-arrival time constant despite changing k, for the Erlang-k distribution we use rate $k\lambda$. To obtain a hyper-exponential with the same mean, we use a mixture of two exponential distributions with parameters λ/k and $k\lambda$, and a probability of $k/(1+k)$ and $1/(1+k)$, respectively. Recall that the coefficient of variation of the

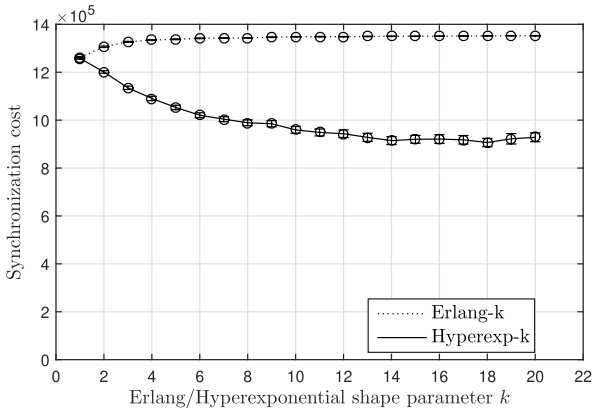

FIGURE 11. Synchronization cost vs. inter-arrival time distribution ($T = 2$, $M_d = 3$).

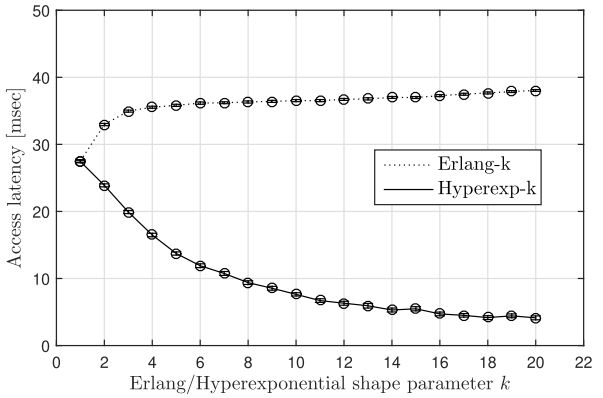

FIGURE 12. Access latency vs. inter-arrival time distribution ($T = 2$, $M_d = 3$).

Erlang-k distribution decreases with k (and thus the Erlang-k resembles the deterministic distribution for large k), while the coefficient of variation of the mixture of two exponential distributions parameterized by k increases with k.

Figure 11 shows the synchronization cost as a function of the distributions' shape parameter k. The results show that the synchronization cost is fairly insensitive to the inter-arrival time distribution. Figure 12 shows the corresponding results for the average access latency. The figure shows that if the arrival process is closer to deterministic than a Poisson process (Erlang-k), the access latency does not increase significantly with k. On the contrary, the access latency drops significantly as the arrival process becomes less deterministic than the Poisson process (hyper-exponential). Thus, the proposed model-based timer selection algorithm would lead to a small violation of the average latency constraint if the arrival process is more deterministic than the Poisson process, and would lead to latencies significantly below the threshold for an arrival process that is less deterministic than the Poisson process.

VII. CONCLUSION

In this paper, we have proposed a fog-assisted aggregated synchronization (FAS) scheme for mobile cloud storage applications. In the FAS scheme, to reduce the synchronization traffic, the fog server and the cloud server conduct

localized and aggregated synchronizations, respectively. To evaluate the performance of the FAS scheme, we derived an analytical model of the FAS scheme and validated it against simulations. The analytical results and extensive simulation results demonstrate the effects of the timer, the update-to-access ratio, the number of mobile users, and the sensitivity to the arrival process. Simulation results show that the FAS scheme can reduce the synchronization traffic significantly with acceptable access latency, and is fairly insensitive to the temporal characteristics of the file access pattern. Our work opens up for a number of interesting research questions concerning mobile edge cloud support for mobile cloud storage, including modeling the impact of mobility on fog-assisted synchronization and the signaling required to handle mobility, and further optimizations that leverage the spatio-temporal characteristics of user access. These question will be subject of our future work.

ACKNOWLEDGMENT

This paper was presented at the IEEE INFOCOM 2014 Student Workshop, Toronto, Canada, April 2014 [1].

REFERENCES

[1] G. Lee, H. Ko, and S. Pack, "Proxy-based aggregated synchronization scheme in mobile cloud computing," in *Proc. IEEE INFOCOM Student Workshop*, Apr./May 2014, pp. 187–188.
[2] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Cloud computing networking: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 54–62, Jul. 2013.
[3] D. Huang, T. Xing, and H. Wu, "Mobile cloud computing service models: A user-centric approach," *IEEE Netw.*, vol. 27, no. 5, pp. 6–11, Sep./Oct. 2013.
[4] Y. Cui, Z. Lai, and N. Dai, "A first look at mobile cloud storage services: Architecture, experimentation, and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 16–21, Jul./Aug. 2016.
[5] M. Akter, A. Gani, M. O. Rahman, M. M. Hassan, A. Almogren, and S. Ahmad, "Performance analysis of personal cloud storage services for mobile multimedia health record management," *IEEE Access*, vol. 6, pp. 52625–52638, Sep. 2018.
[6] T. Noor, S. Zeadally, A. Alfazic, and Q. Z. Sheng, "Mobile cloud computing: Challenges and future research directions," *J. Netw. Comput. Appl.*, vol. 115, pp. 70–85, Aug. 2018.
[7] *Microsoft Makes Office Mobile Editing Free, Launches Separate iPhone Apps and Preview for Android Tablets*. Accessed: May 4, 2019. [Online]. Available: http://venturebeat.com/2014/11/06/microsoft-makes-office-mobile-editing-free-launches-separate-iphone-apps-and-preview-for-android-tablets/
[8] *Collaboration Just Got Easier: Real-Time Co-Authoring Now Available in Office Web Apps*. Accessed: May 4, 2019. [Online]. Available: http://blogs.office.com/2013/11/06/collaboration-just-got-easier-real-time-co-authoring-now-available-in-office-web-apps/
[9] Z. Li *et al.*, "Efficient batched synchronization in dropbox-like cloud storage services," in *Proc. Middleware*, 2013, pp. 307–327.
[10] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking personal cloud storage," in *Proc. ACM IMC*, 2013, pp. 205–212.
[11] Z. Li *et al.*, "Towards network-level efficiency for cloud storage services," in *Proc. ACM IMC*, 2014, pp. 115–128.
[12] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: Understanding personal cloud storage services," in *Proc. ACM IMC*, 2012, pp. 481–494.
[13] G. Wu *et al.*, "On the performance of cloud storage applications with global measurement," in *Proc. IEEE IWQoS*, Jun. 2016, pp. 1–10.
[14] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2013.
[15] Z. Li, Z.-L. Zhang, and Y. Dai, "Coarse-grained cloud synchronization mechanism design may lead to severe traffic overuse," *Tsinghua Sci. Technol.*, vol. 18, no. 3, pp. 286–297, Jun. 2013.
[16] N. Samteladze and K. Christensen, "DELTA: Delta encoding for less traffic for apps," in *Proc. IEEE LCN*, Oct. 2012, pp. 212–215.
[17] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.
[18] B. Calder *et al.*, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. ACM SOSP*, 2011, pp. 143–157.
[19] W. Hu, T. Yang, and N. J. Matthews, "The good, the bad and the ugly of consumer cloud storage," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 110–115, 2010.
[20] H. Slatman, "Opening up the sky: A comparison of performance-enhancing features in skydrive and dropbox," in *Proc. 18th Twente Student Conf. IT*, 2013, pp. 1–8.
[21] J. P. Ramos, L. Veiga, and P. Ferreira, "vfcBOX: Multi-user consistent file sharing," in *Proc. MGC*, 2011, p. 5.
[22] J. E, Y. Cui, P. Wang, Z. Li, and C. Zhang, "CoCloud: Enabling efficient cross-cloud file collaboration based on inefficient Web APIs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 56–69, Jan. 2018.
[23] Y. Cui, Z. Lai, X. Wang, and N. Dai, "QuickSync: Improving synchronization efficiency for mobile cloud storage services," *IEEE Trans. Mobile Comput.*, vol. 16, no. 12, pp. 3513–3526, Dec. 2017.
[24] G. Lee, H. Ko, and S. Pack, "An efficient delta synchronization algorithm for mobile cloud storage applications," *IEEE Trans. Service Comput.*, vol. 10, no. 3, pp. 341–351, May/Jun. 2017.
[25] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," ETSI, White Paper 11, Sep. 2015.
[26] Y. Hao *et al.*, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.
[27] M. Miller, *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. Indianapolis, IN, USA: Que Publishing, 2009.
[28] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 989–997, May 2014.
[29] S. Chen *et al.*, "When queueing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1042–1050.
[30] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Pérez, "Consistency in the cloud: When money does matter!" in *Proc. IEEE/ACM CCGrid*, May 2013, pp. 352–359.
[31] S. M. Ross, *Stochastic Processes*. New York, NY, USA: Wiley, 1996.
[32] J. Xie and I. E. Akyildiz, "A distributed dynamic regional location management scheme for mobile IP," *IEEE Trans. Mobile Comput.*, vol. 1, no. 3, pp. 163–175, Jun. 2002.
[33] G. Lee, I. Jang, S. Pack, and X. Shen, "FW-DAS: Fast wireless data access scheme in mobile networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 8, pp. 4260–4272, Aug. 2014.

GIWON LEE received the B.S. and Ph.D. degrees from Korea University, Seoul, South Korea, in 2009 and 2015, respectively. Since 2015, he has been with Samsung Electronics, Gyeonggi, South Korea. His research interests include mobile cloud computing, software-defined networking, artificial intelligence, 5G networks, and the future Internet.

HANEUL KO received the B.S. and Ph.D. degrees from the School of Electrical Engineering, Korea University, Seoul, South Korea, in 2011 and 2016, respectively. He is currently an Assistant Professor with the Department of Computer Convergence Software, Korea University, Sejong, South Korea. From 2016 to 2017, he was a Postdoctoral Fellow of mobile networks and communications, Korea University, Seoul, South Korea. From 2017 to 2018, he was with the Smart Quantum Communication Research Center, Korea University, Seoul, South Korea, and a Visiting Postdoctoral Fellow of the University of British Columbia, Vancouver, BC, Canada. His research interests include 5G networks, mobility management, mobile cloud computing, SDN/NFV, and the future Internet.

SANGHEON PACK received the B.S. and Ph.D. degrees in computer engineering from Seoul National University, Seoul, South Korea, in 2000 and 2005, respectively. He joined Korea University, Seoul, South Korea, in 2007, as a Faculty Member, where he is currently a Professor with the School of Electrical Engineering. From 2005 to 2006, he was a Postdoctoral Fellow of the Broadband Communications Research Group, University of Waterloo, Waterloo, ON, Canada. His research interests include the future Internet, SDN/ICN/DTN, mobility management, mobile cloud networking, multimedia networking, and vehicular networks. He was a recipient of the IEEE ComSoc APB Outstanding Young Researcher Award, in 2009, the KICS Haedong Young Scholar Award, in 2013, the Korea University TechnoComplex (KUTC) Crimson Professor, in 2015, the IEEE/IEIE Joint Award for IT Young Engineers Award, in 2017, and the KIISE Young Information Scientist Award, in 2017. He will serve as the TPC Vice-Chair of information systems of the IEEE WCNC 2020, and has served the Track Chair of the IEEE CCNC 2019. He has served as the Co-Chair of the IEEE VTC 2010-Fall Transportation, the Publicity Co-Chair of the IEEE SECON 2012, the Co-Chair of the IEEE WCSP 2013 Wireless Networking Symposium, the TPC Vice-Chair of ICOIN 2013, the Publication Co-Chair of the IEEE INFOCOM 2014, ACM MobiHoc 2015, and the Track TPC Chair of EAI Qshine 2016. He is an Editor of the IEEE Internet of Things (IoT) journal, the *Journal of Communications Networks* (JCN), and *IET Communications*. He is a Guest Editor of the IEEE Transactions on Emerging Topics in Computing (TETC).

VALENTINO PACIFICI received the M.Sc. degree in computer engineering from the Politecnico di Milano, and the M.Sc. and Ph.D. degrees from the KTH Royal Institute of Technology, Stockholm, where he is currently a Postdoctoral Researcher with the Laboratory of Communication Networks. His research interest includes key questions in the areas of content management systems.

GYÖRGY DÁN received the M.Sc. degree in computer engineering from the Budapest University of Technology and Economics, Hungary, in 1999, the M.Sc. degree in business administration from the Corvinus University of Budapest, Hungary, in 2003, and the Ph.D. degree in telecommunications from KTH, in 2006. He is currently an Associate Professor with the KTH Royal Institute of Technology, Stockholm, Sweden. He was a Consultant in the field of access networks, streaming media, and videoconferencing, from 1999 to 2001. He was a Visiting Researcher with the Swedish Institute of Computer Science, in 2008, a Fulbright Research Scholar with the University of Illinois at Urbana–Champaign, from 2012 to 2013, and an Invited Professor with EPFL, from 2014 to 2015. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security in power systems.